Data Analysis, Statistics, Machine Learning

Leland Wilkinson

Adjunct Professor
UIC Computer Science
Chief Scientist
H2O.ai

leland.wilkinson@gmail.com
What is (are) data?

A datum is a given (as in French donnée)
   data is plural of datum

Data may have many different forms
   Set, Bag, List, Table, etc.
   Many of these forms are amenable to data analysis
   None of these forms is suitable for statistical analysis

Statistics operate on variables, not data
   A variable is a function mapping data objects to values
   A random variable is a variable whose values are each associated with a probability $p$ ($0 \leq p \leq 1$)

Visualizations operate on data or variables
Datasets

A set cannot contain duplicate elements

On IBM mainframes, datasets originally had a record structure

  So each record is an element
  But duplicate records can occur
  So IBM mainframe datasets were really lists (ordered sets allowing duplicates)

Nowadays, the term dataset is interpreted much more broadly

  Flat files
  Relational tables
  Distributed databases
  Graph databases
    Object databases were a precursor
    Objects can be images, emails, etc.
  Streaming (real time) databases
Data

Flat files

Flat file databases (e.g., FileMaker) store their data in tables
Flat file tables have $n$ rows and $p$ columns
The rows are independent of each other (no relations)
The CSV file (Microsoft Excel export) is most popular text format
  One row per record
  Fields are separated by comma, blank, semicolon, vertical bar, etc.
  There is no standard for CSV (like an IEEE standard), unfortunately
  Thus, it’s an art to import them

Statistics packages (BMDP, SAS, SPSS) originally used this format
  Each row corresponds to a case or observation or sampling unit
  Each column (field) corresponds to a variable
Stat packages eventually moved to binary encoding of their files
But CSV text is often more compressed (especially for floats)
So stat package binaries are becoming less common
Data

Importing Text Files (including CSV)

<table>
<thead>
<tr>
<th>Separators:</th>
<th>,</th>
<th>\t</th>
<th>Blank(s)</th>
<th>;</th>
<th>:</th>
<th></th>
<th>and a few other rare ones</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lines per record:</td>
<td>one</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Missing values:</td>
<td>&quot;&quot;</td>
<td>\t \t</td>
<td>;;</td>
<td>::</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Header:</td>
<td>optional (rows may contain fewer values than header)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Strings:</td>
<td>strings may be surrounded by quotes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>They MUST be surrounded by quotes if they contain a separator or blanks</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Embedded quotes represented by &quot;&quot;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

John, Doe, 120 Jefferson St., Riverside, NJ, 08075
Jack, McGinnis, 220 Hobo Av., Phila, PA, 09119
"John ""Da Man"", Repici, 120 Jefferson St., Riverside, NJ, 08075
Stephen, Tyler, "7452 Terrace ""At the Plaza"" road", SomeTown, SD, 91234
Blankman, , SomeTown, SD, 00298
"Joan ""the bone"", Anne", Jet,"9th, at Terrace plc", Desert City, CO, 00123
Data

Relational databases

Edgar Codd invented the relational database

IBM refused to implement it until Codd shamed them

According to Codd, a RDBMS cannot contain duplicate rows

But SQL allows it, much to Codd’s annoyance

The Codd model

A relation variable is a function relating values to objects

A relation variable has a domain

A domain is a set of scalar values of one type (integers, strings, etc.)

A “table” is a metaphor for a collection of relations

Each “row” of a table is a tuple

Each “column” is an attribute specifying a domain

Quotes are used here because tables are not relations

Tuples are unordered

Attributes are unordered
Data

SQL (Sequential Query Language)

SQL was designed to manipulate tables
Tables are not relations

Some versions of SQL violate the relational model

SQL doesn’t have a universally recognized specification
Tuples in the relational model are unordered sets of known values
But SQL doesn’t talk tuples. It talks rows.

A row is an ordered set of known or unknown values with names
A row can have NULLs. A tuple cannot.
Duplicate rows are permitted. Duplicate tuples are not.

The consequence of all this is that SQL restricts us to a table model
And we have to customize our SQL to match a provider’s standard
Data

Analyses on Relational databases

We can do data analysis on tables

But there are problems doing statistics on tables

Many statistical methods presume a vector space
Each vector is an ordered tuple of real numbers
  In the relational model, elements of a tuple are unordered
Time series methods assume rows are ordered (in time)
  Not allowed in the relational model
A sample can contain duplicate rows (cases, observations)
  Not allowed in the relational model

There are hacks to fix these problems

Add an index variable to allow duplicate rows
Impose an ordering on the columns of the table (using SQL)
Impose an ordering on the rows through an index or date variable
Analyses on Relational databases

Aggregation

Aggregate variables may not be independent across rows
Aggregate variables are not necessarily normally distributed
  - Counts are not normally distributed
  - Sums and means of normal variables are normal
  - Sums and means of many other variables are normal if \( n \) is large
  - But variances and covariances of aggregates may pool heterogeneous sources

**Ecological fallacy** can arise with inferences based on aggregation
  - Correlations at the group level can be much higher than those at the individual level
  - Aggregate behavior is not the same as individual behavior
  - There is no easy fix for this problem except to avoid aggregation in these cases

Hierarchical models are designed to deal with this problem
  - But they are not practical in SQL

**Simpson’s paradox** can jeopardize inference based on aggregates
  - The story told by the aggregate is contradicted by the one told by the disaggregates
Data

Analyses on Relational databases

Simpson’s Paradox

Disaggregate

Aggregate
Data

Analyses on Relational databases

Database statistical analytics

Some companies may not program statistics correctly

- Database companies do not always have computational statisticians on staff
- For example, numerous publications have revealed inaccuracies in Excel statistics
- Accurate algorithms for even simple estimates (e.g., means) are not found in textbooks
- Bottom line: trust entities whose business is statistics to get correct results

Missing values often not handled correctly in database statistical algorithms

- Databases represent missing values as NULL
- NULL can mean “Does not apply” or “Don’t know” or “Refused to reply” etc.
- Built-in algorithms may not process these appropriately (e.g., multiple imputation)

One solution (used by SAS and R) is to embed appropriate calculations in database

- Some databases accept C or Java embeddings under read/write restrictions
- But database administrators HATE this

Copyright © 2016 Leland Wilkinson
Data

Analyses on Relational databases

Why can’t we do inferential statistics on sums?

Sum depends on $n$

- Sums of random variables do have a distribution
  - Which is normal-like if $n$ is large
  - But they need to be identically distributed random variables
  - And we need to fix $n$ to assume the distribution.

We need a variance estimate

- We need to have a variance estimate, not a single sum, in order to assess risk
  - In certain circumstances (e.g., baseline variance for arcsine transform) we can do this, but not often

The entities on which the sums are based need to be comparable

- These are not comparable
  - Total sales over superstores of different sizes and locations
  - Budgets of states of different populations
  - Total revenue of Microsoft, Facebook, Apple, Google, and Twitter

- These are comparable
  - Sum of scores on individual items of a well-designed test

Inferences based on sums do not apply to their constituents

- Inferences on classroom test scores cannot be applied to students
Data

Distributed File Systems

The problem

In memory and classic databases cannot handle huge files
Distributed databases can but there are other problems involving complexity
Statistical calculations on single files do not scale well

Map Reduce (Hadoop)

This Google invention distributes data over many cores
By slicing up data, many (but not all) statistical calculations can run in parallel
Hadoop is the open-source version of Map Reduce

PSM (Pass Stream Merge)

Unlike Map Reduce, there is no central supervisor or hierarchy

Biological analogy

Lots of amoebas swimming around in cloud soup
Localizes three methods (Pass, Stream, Merge) in one class
Data

Map Reduce (Hadoop)

Fault tolerance
I/O Scheduling
Automatic parallelization and distribution
Hadoop has hundreds of third-party vendors
Works for many algorithms
Slower than in-memory systems for some algorithms
Map Reduce (Hadoop)

Data slice → Map → Intermediate → Reduce → Final Result

Distributed data

Intermediat data
PSM (Pass Stream Merge)

Analytic

Summaries

Summary

Statistics

Statistic

Sums

Means

Ranges
Data

PSM (Pass Stream Merge)

A StatTree is a tree of Tables.
A Table is a table of data.
An input dataset is a Table.
An externalized Statistic is a Table.
Analytics input and output StatTrees.
Data

PSM (Pass Stream Merge)

PASS

Iterator iter = input.getIterator();
while (iter.hasNext()) {
  Row row = (Row) iter.next();
  double weight = row.getWeight();
  double[] data = row.getContinuousData();
  if (weight > 0) {
    for (int j = 0; j < data.length; j++) {
      double xd = weight * (data[j] - means[j]);
      double xj = data[j];
      counts[j]++;
      sums[j] += weight * xj;
      means[j] += xd / wcounts[j];
    }
  }
}
PSM (Pass Stream Merge)

STREAM

```java
Iterator iter = input.getIterator();
while (iter.hasNext()) {
    Row row = (Row) iter.next();
    double weight = row.getWeight();
    double[] data = row.getContinuousData();
    if (weight > 0) {
        for (int j = 0; j < data.length; j++) {
            double xd = weight * (data[j] - means[j]);
            double xj = data[j];
            if (add) {
                counts[j]++;
                sums[j] += weight * xj;
                means[j] += xd / wcounts[j];
            } else {
                counts[j]--;
                sums[j] -= weight * xj;
                means[j] -= xd / wcounts[j];
            }
        }
    }
}
```
public void merge(Summary s1, Summary s2) {
    double[] means1 = Means.extractData(s1);
    double[] means2 = Means.extractData(s2);
    double[] wcounts1 = WeightedCounts.extractData(s1);
    double[] wcounts2 = WeightedCounts.extractData(s2);
    for (int i = 0; i < means1.length; i++) {
        means1[i] = (wcounts1[i] * means1[i] + wcounts2[i] * means2[i]) / (wcounts1[i] + wcounts2[i]);
    }
    Means.setData(s1, means1);
}
Data

Distributed File Systems

What they are good for
Additive algorithms (ordinary regression, logistic regression, ...)

What they are bad for
Inefficient for nested iterative and interactive analytics
Nested iterative algorithms require multiple map/reduce steps
This propagates multiple files between MapReduce phases
And file processing is slow, even for in-memory files
Can be slower on some problems than single-thread table processing
The Hadoop hype conceals these problems
No OS can implement parallel processing for all algorithms automatically
For some algorithms, the only way is to customize parallelization
Graph databases

A graph $G = (V,E)$ is a pair of sets

- $V$ is a set of vertices (sometimes called nodes)
- $E$ is a set of edges (sometimes called arcs or links)
- $E$ induces a relation on $V$

The simplest file representation of a graph is a edge list text file

- Each row is an edge
- A program can parse this file to construct a list of nodes and a list of edges

Graph databases store nodes indexed by edges

- Each node (an object) has a set of properties
- Queries on graphs are especially efficient for traversal, etc.
- Statistical analytics (degrees, centrality, diameter, etc.) are straightforward
- Scalable (sparseness is built in to the schema)
Data

Streaming (real time) databases

Data objects ordered in time
Data objects enter in real time
  Stock market series
  Instrumented data
  Business metrics
Data objects retrievable with short delay
Statistical methods must be designed to handle streaming inputs
  Running window (moving average, etc.)
  Update/downdate methods
The PASS/STREAM/MERGE algorithm was designed to deal with these
Data

Symmetric matrices

S is a symmetric matrix \(s_{ij} = s_{ji}\)

Types of data that comprise symmetric matrices

- Similarities
- Dissimilarities
- Distances
- Correlations
- Derived distances

Analytic methods whose input is a symmetric matrix

- Multidimensional Scaling (MDS)
- Principal Components
- Hierarchical Clustering
Data

Symmetric matrices

Start with a matrix $X$ and transform elements into a single column

\[
\text{Sloane sequences database}
\]
Data

Symmetric matrices

Start with a matrix $X$ and transform elements into a single column

Expected Value for Normal Distribution

Copyright © 2016 Leland Wilkinson
Structured matrices
Sometimes data matrices are not i.i.d.

Simplex  Band  Circumplex  Equi  Block
Sequences

A dataset may be an ordered list or a set of ordered lists

Sequence analysis